Portfolio Report

Design, create, and debug dynamic web pages using a client-side language.

In developing the software, HTML was mainly used to structure the web pages. CSS and Bootstrap were used to style the website. The main style of the website is uniform, and all have the same layout. Because this is a fitness software, users might use it on mobile devices. Bootstrap is a responsive layout design that allows the website to be compatible with devices of different resolutions. So Bootstrap is being used primarily for styling. JavaScript, jQuery and Ajax are used to interact with the backend to implement various functions in the web pages such as passing and saving data, jumping pages etc.

```
$(function () {
    // restart button
    $("input[name='restart']").click(function () {
        window.location.href="/DetailsInfo";
    });
    // save button
    $("input[name='save']").click(function () {
        var sex = $("input[name='sex']").val();
        var height = $("input[name='height']").val();
        var weight = $("input[name='weight']").val();
        var level = $("input[name='level']").val();
        var speciality = $("input[name='speciality']").val();
        var qualification = $("input[name='qualification']").val();
        var organisation = $("input[name='organisation']").val();
        if(sex==""||height==""||weight==""||level==""||speciality==""||qualification==""||organisation==""){
            dialog("cannot empty! ");
            return;}
        var data = {"sex":sex,"height":height,"weight":weight,"level":level,
            "speciality":speciality,"qualification":qualification,"organisation":organisation};
        update_Info(data);
    });
})
```

```
function update_Info(data) {
    $.ajax({
        method:"POST",
        type:"POST",
        url:"/updateInfo",
        data:data,
        success:function (msg) {
            if(msg=="success"){
                dialog("success");
            }else{
                dialog("failed");
            }
        },error:function () {
            dialog("ajax wrong");
        }
    });
}
```

Figure 1

For example, as shown in Figure 1. On a web page that updates client information, a function written in jQuery saves and checks the information entered by the user on the web page, and if there are no null values, calls the function update-Info to update the client information. If there have null value, a warning pops up. jQuery is a JavaScript-based front-end framework. For this functionality, jQuery can fetch user input with clean code, achieving the same functionality with less code than fetching data directly with JavaScript. Since jQuery encapsulates all Ajax operations into the *$.ajax()* function, this makes development easier.

The main tool for debugging client-side code is Chrome. Element can be viewed to modify the elements on the page. I mainly use it to debug CSS code, as it is a team project, so there may be style conflicts, Element can show these conflicts very well and can check or uncheck the relevant styles, which is very helpful for web debugging. The error message can then be displayed in the Console. I regularly check for relevant error messages in the console. Breakpoint debugging allows problems to be pinpointed as they arise. If during debugging something needs to be changed, it can be changed and saved directly using Chrome, rather than having to go back to the code and change it and run it again.

In my opinion, I have done a good job of developing the structure, style and data interaction with the backend on the client side. The uniformity of pages should be further improved in the future by using Thymeleaf to implement common headers and footers on a number of different pages.

Design, create, and debug server-side code and utilise this from dynamic web pages.

In the development of the server side.

```
import javax.persistence.*;
@Entity
public class Comment {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;
    private String content;
```

Figure 2

The persistence class was created to implement the entity class. (as shown in Figure 2) Use *@Entity* to mark the persistence class and the Spring Boot framework will automatically build a table based on the persistence class when it is loaded. Use *@Id* to specify the primary key. *@GeneratedValue(strategy=GenerationType.AUTO)* is used to specify the generation strategy for the primary key, which defaults to self-growing in MySQL.

```
import org.springframework.data.repository.CrudRepository;

public interface CommentRepository extends CrudRepository<Comment, Integer> {
```

Figure 3

To implement database *CRUD*, I created CommentRepository. The CommentRepository class is then used to define the data access interface, which extends the CrudRepository interface (as shown in Figure 3) with the first type parameter being the persistent object Comment, indicating the type of persistent object currently being manipulated, and the second type parameter being an Integer, specifying the ID type.

```
@RequestMapping(path = "/addComment")
ModelAndView addNewComment(@RequestParam String content,
                           Comment comment) {
    comment.setContent(content);
    commentRepository.save(comment);
    log.info(comment.toString() + "save to the repo");
    return new ModelAndView( viewName: "forward:/html/consoleClient.html"); }
```

Figure 4

The front-end pages are connected to the server-side code via the Controller. (as shown in Figure 4) Use @RequestMapping to map requests, i.e., to specify which URL requests the Controller can handle. The matching URL executes the corresponding function, calling the properties of the pre-declared CommentRepository to perform operations on the database. Then the return value of the Controller is forward to the requested page.

It is worth mentioning the use of Java multithreading in the server-side code to increase the user experience. Because the user will receive an email with personal information after registration, the process of sending the email is a bit slow and would result in extra waiting if multi-threading was not used. I think this is a good idea.

To debug the server-side code, I use IntelliJ IDEA as a debugging tool. I start the application in debug mode and add breakpoints to the parts of the code that I would like to debug. This is very handy for pinpointing the location of errors.

Compared to using JdbcTemplate, Spring Data JPA no longer needs to write SQL statements, which greatly simplifies JPA persistence by writing repository interfaces and managing data through various annotations on entity classes.

I think the development of the server side is relatively well developed. In the future there is a need to improve the uniformity of the server-side code and to make full use of the encapsulated tool classes.

Design and create a database that integrates with web pages.

	Field	Type	Null	Key	Default	Extra
▶	id	int(11)	NO	PRI	NULL	
	content	varchar(255)	YES		NULL	

Figure 5

The simpler the better when designing a database, the simplest data type possible is used. To try to avoid null values, I set the id as primary key in each database to be int type and not null and set it auto-increment. For string data, I set it to varchar and set the length to 255. There are generally two types of strings in databases, varchar and char. Compared to char, varchar holds variable length strings and is the most used string type, it can take up less storage space than fixed types because it only takes up as much space as it needs.

The reason why MySQL was chosen as the project database was because it is an open-source database system that is small, easy to manage and suitable for small to medium sized projects. Compared to SQLite, MySQL supports more data types, and it is also easier to use. Thus, this project choice MySQL. We put the designed SQL script file in SpringBoot so that the SQL file can be executed automatically when the project is started, for the purpose of creating the database and data tables.

Integrating the web pages with the database requires importing the dependencies and configuration in the configuration file. Entity classes are then written to extend the methods in the CrudRepository to connect to and operate with the database using the repository interface written to it. The entity classes are used to work with the methods in the repository for the purpose of manipulating the database. Finally, the methods are called in the Controller file to complete the interaction with the front-end interface.

The tables designed in the project are very basic and I think the design of these tables is well done. What needs to be improved in the future is the management of the database in the collaboration of team projects.

Demonstrate a systematic understanding of computing concepts relating to Web applications, both theoretical and practical.

For the URL submission and data transfer, I use the *$.ajax()* method to implement asynchronous data transfer after the underlying data encapsulation via the HTTP protocol in update client profile page. This submission method brings the user experience closer to that of a desktop application. Submitting via hyperlinks or forms, on the other hand, requires jumping between pages to submit information, sacrificing the user experience. Compared to hyperlinks and forms, this submission method uses a partial page refresh, where the user can refresh the data on this page, thus improving the user experience.

By setting the form in the web interface to submit as POST, the data is submitted to the specified resource and requested to be processed by the server. The data is included in the request text. The other common request method, GET, is not used because the GET method should only be used to read data and should not be used in operations that produce "side effects", such as GET being accessed by web spiders. This is because GET includes the parameters in the URL while POST passes them through the request body.

I was guided by the principles of short, descriptive and efficient when designing the relevant URLs. The main function of the page is expressed in the URL. For example, if user want the browser to display the home page of the current project, users can simply type `Localhost://8080/home` to access it. This improves the ease of access to the site for users.

Spring Security has also been added to this project to increase security. Specify custom requirements for URLs by adding multiple subsections to the *http.authorizeRequests()* method. Allow access to authenticated URL via the *http.anyRequest().authenticated()* method.

I believe that I have done a reasonable job in transferring data between the front and back ends, not only using ajax and jQuery to transfer data, but also using the *window.location.href* in JavaScript to replace hyperlinks in the page jumping aspect. This is easier to maintain and modify. I think it is important to improve my understanding of the framework in the future, to make the web application more integrated and logical, and to add comments to make it easier for other developers to understand the methods.

Critically analyse architectural design patterns relevant to web applications and reflect on choices made.

SpringBoot is based on the Spring Framework, which simplifies the creation and development of applications. SpringBoot was chosen because this project is a team work, and everyone's part of the development was different and there was a chance that some external libraries would be introduced. If SpringBoot is not used there is a risk of library conflicts or the application not running on other machines. SpringBoot works well with Gradle or Maven to inject dependencies so that the team could work together more efficiently.

The MVC model, which is popular in web projects today, is used in this software. The use of the MVC design pattern has brought many benefits to our group's project development, such as reducing program coupling, increasing code reusability, reducing the cost of developing programs and interfaces, and providing greater ease of deployment and maintenance through this layered structure. The disadvantage of MVC in this project is that it is not clearly defined and is not well suited to small to medium sized applications such as this one. It also adds to the complexity of the system structure and implementation, with too tight a connection between the view and the controller. I added jQuery as a mediator between the View and the Controller as appropriate to improve program maintenance.

SpringBoot MVC basically consists of Controller + Thymeleaf (View) + Model, (as shown in Figure 8) Controller plays the role of interface between View and Model, Thymeleaf is View, Model has a special class to correspond with it.

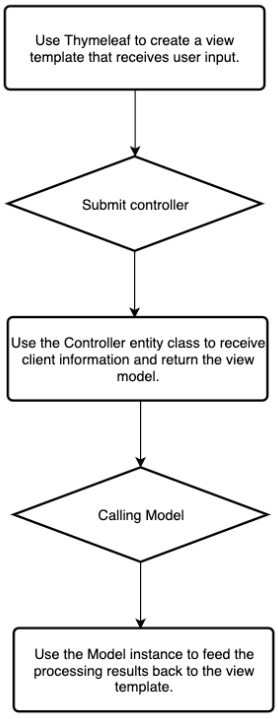

Figure 8

In summary, I think the SpringBoot framework has made creating applications based on the MVC model particularly easy and has greatly improved the efficiency of project development. The project has done a good job in applying the MVC model, and the three layers are clearly outlined. I think the reflective report will help to improve my knowledge of the programming and the understanding of the development I have done. When developing future projects, I need to improve my understanding of a framework and programming patterns. This will make the project development more motivating.